(unknown project)

What is TEI conformance, and why should you care?

Lou Burnard

2017-05-10

Abstract

The recommendations of the Text Encoding Initiative (TEI) seem to have become a defining feature of the methodological framework of the Digital Humanities, despite recurrent concerns that the system they define is at the same time both too rigorous for the manifold variability of humanistic text, and not precise enough to guarantee interoperability of resources defined using it. In this talk I question the utility of standardization in a scholarly context, proposing however that documentation of formal encoding practice is an essential part of scholarship. After discussion of the range of information such documentation entails, I explore the notion of conformance proposed by the TEI Guidelines, suggesting that this must operate at both a technical syntactic level, and a less easily verifiable semantic level. One of the more noticeable features of the Guidelines is their desire to have (as the French say) both the butter and the money for the butter; I will suggest that this polymorphous multiplicity is an essential component of the system, and has been a key factor in determining the TEI's continued relevance.

1. What are encoding standards actually for?

As the old joke says, the good thing about standards is that there are so many to choose from. You can choose to follow a dictatorial, centrally-imposed, we-know-what's-best-for-you encoding method like using Microsoft Word. You can choose to follow a hand-crafted, idiosyncratic, we-know-what-we're-doing kind of encoding standard made up and maintained by the leading lights of a particular research community, like Epidoc. Or you can just go ahead and do your own encoding thing, which I like to characterize as the nobody-understands-my-problems kind of standard. In academia, there's a good argument for each of these flavours. WKWBFY saves a lot of time and effort reinventing the wheel and ensures that your work will be processable and usable in at least one kind of application environment: the downside is that you may not want or like the world view that the system embodies, but you can't change it. WKWWD probably means you are dealing with congenial and familiar views and are guaranteed respect within your community, but no-one outside that community will know what to do with your stuff, and you may be a bit limited if you want to push the boundaries of knowledge or praxis within it. And, of course, NUMP guarantees you the luxury of making all your own decisions, getting everything just the way you want, but consequently not only risking isolation from your peers but also having to spend lots of time and effort doing tecchie things that have nothing to do with your real scholarly preoccupations.

When the choice is so hard to make, it may be a good idea to reconsider the motivation for making it in the first place. What do we gain from adopting an explicit encoding standard? What scholarly (as opposed to social, if the two are distinguishable) advantage do we gain in defining formally the methods and formats by which we choose to represent in digital form our understanding of cultural artefacts? We may do it simply in order to be seen to be ticking the right boxes in a funding agency's list of criteria; we may do it because our elders and betters have told us we should; we may do it because we know no better. But none of these can be considered well-founded motivations. How does the use of explicit standards in the markup of digital resources contribute to the success or failure of a scholarly enterprise based on the use of such resources?

Firstly, I suggest, we should not forget that the application of markup is an inherently scholarly act: it expresses a scholarly interpretation. It is a hermeneutic activity. Our choice of markup vocabulary is therefore not an arbitrary one. It has consequences. It may make it harder to conceive of the truth about a document or a document's intentions; it may make it much easier to say something which is convenient, but false. To dismiss as ‘mere semantics’ concerns about the proper application of markup is thus to embark upon a very dangerous path, assuming that you share my belief that every scholarly encoding should truthfully represent without convenient distortion a scholarly reading. I make no claim here for the absolute truth or otherwise of the interpretation itself : that is, as they say, beyond my pay grade.

Secondly, if the function of markup is to express an interpretation, then the markup language itself should as far as possible eschew ambiguity. Markup defines and determines the interface between algorithmic processing and human interpretation. Life is complicated enough without introducing additional fuzziness and inconsistency into the processing stack. We would like to live in a world where two equally well informed observers looking at the same encoding will reach similar or identical conclusions as to the interpretations which occasioned that encoding. We would also like to be confident that two equally well-informed encoders, considering the same textual phenomenon, and having the same interpretation of it, will encode that interpretation in the same way. (This is not, of course, the same as wishing that all well-informed encoders should reach the same interpretative conclusions about a given text. Quite the contrary.) Consequently, as far as possible, we expect the claims embodied by a marked up document to be formally verifiable in some way. Verifiability implies the existence of some formal definition for the markup language, against which productions using it can be checked, preferably using an automated system such as a parser or other automaton. Talking solely of XML documents, it is commonplace to require that the concept of ‘well-formedness’ -- purely syntactic correctness -- be completed by the concept of ‘validity’ -- conformance to a specific XML markup vocabulary, as defined by a schema of some sort.

Scholarly markup however requires more than simple XML validity. It may be necessary to do more than constrain the context or name of a given markup component; indeed we might argue that it is always necessary to do so. The surface components of a marked up document (the start and end tags, the attribute value pairs etc.) have intention beyond what an XML schema can express. A typical XML schema will allow me to say that the XML element `<p>` must appear within the XML element `<div>` and not the reverse, but it won't easily let me say that the content of my `<p>` elements should correspond to a paragraph of text rather than, say, a list item, heading, or page in the source document being encoded. For that information, I will need to consult the project-specific documentation, which should spell out how exactly the intentions behind this set of encoded documents.

Thirdly, therefore, we need to complement the automatic verifiability of our markup with semantic controls which, in our present state of knowledge, are not automatable, and require human judgment. It is no coincidence that SGML, the ancestor of XML, was produced by a lawyer: the rules embodied by an SGML DTD, like those in the statute book, must be interpreted to be used. In the field of the law, the statute book is completed by precedents; in the case of an XML schema used by a broad community such as the TEI, the rules incarnated in the TEI Guidelines must be completed by practice of those using them, whether we are thinking about the Guidelines as a whole, or the customizations of them used by individual projects. A TEI customization expresses how a given project has interpreted the general principles enumerated by the Guidelines, as well as formally specifying which particular components of the Guidelines it uses. It also provides ample opportunity, through documentation and exemplification, to guide a human judgment as to the way in which the markup should be understood, and therefore the extent to which different datasets using it can be integrated or rendered interoperable, a point to which we will return.

2. How are encoding standards to be documented?

As a minimum, the documentation of an encoding language has to be able to specify the same things as a schema does: the names of the elements and attributes used, their possible contents, how elements may be validly combined, what kinds of values are permitted for their attributes, and so on. The schema languages currently available to us do not provide an entirely identical range of facilities of this kind, nor do they conceptualise the validation of documents in exactly the same way, but they are in sufficiently broad agreement for it to be possible to model the information they require using a simple XML language.[i]

However, if schema models were all that the TEI tag documentation system supported, it would be hard to persuade anyone to use it. The full ODD language, as this tag set is known[ii], allows a rich and well organized specification to be created for individual elements and attributes, for datatypes, for classes, and for macros (i.e. strings), in each case providing much more than the basic information required to create a schema using them. The TEI itself is composed of several hundred such specifications, from which are generated formal schemas and the two volumes of reference documentation which we no longer print, but instead provide online at http://www.tei-c.org/Guidelines/. To use the TEI, one also selects from this set of specifications, optionally modifying or extending its components, to generate a customized schema with accompanying documentation.

A full TEI element specification contains:

- a *canonical name* for the element, together with (as necessary) explanatory glosses for the name in various languages; alternative names in other languages; equivalents in other markup schemes;
- at least one summary *description* of the meanings and usages intended for the element;
- information about the element and attribute *classes* to which the element belongs;
- information about the element's *content model* as already indicated;
- formal specifications for any *constraints* additional to those expressed by the content model;
- a list of specifications for any *attributes* defined as local to the element rather than being inherited from an attribute class;
- formal specifications for the recommended *processing model* applicable to the element;
- annotated *examples* of usage;
- additional *commentary* or usage notes;
- a list of *references* to the chapter of the Guidelines text where the element is discussed more fully.

Without going into too much detail, it should be evident that a TEI specification therefore potentially provides data which can be used to facilitate many different markup processes in a more intelligent way, making it possible for example to provide context sensitive help in different languages, to generate formal schema specifications in different schema languages automatically, to provide user-oriented tutorial manuals, to develop more intelligent data entry and validation systems, or simply to act as a point of entry into the canonical TEI documentation.

In its most recent form, a TEI element specification even attempts to address a criticism sometimes made of XML systems in general and the TEI in particular: that their focus on data independence leads to a focus on the platonic essence of the data model at the expense of an engagement with the rugosities needful when making the data actually useful or usable. The ‘processing model’ mentioned above is a recent addition to the TEI ODD language intended to redress that balance a little: it allows for a more formal specification of the kind of processing that the encoder considers appropriate for a given element. At its simplest, this might indicate the class of formatting appropriate for it; or it might indicate that this is one of the so-called ‘janus’ elements which present alternative encodings for the same phenomenon. In either case, the intention is to simplify the task of the application developer faced with a specific customization of the TEI.

A TEI customization is made by selecting from the available specifications. To facilitate that task, the specifications are grouped together both physically into named ‘modules’, and logically into named ‘classes’. Each module contains a varying number of related declarations. In earlier versions of the Guidelines, a distinction was made between modules which provided components specific to a particular kind of document (the ‘base’ tagsets) and those which provided components specific to a particular kind of analysis (the ‘topping’ tagsets). The idea was that a schema would typically use a single base and multiple toppings, though it was also possible to combine multiple bases. This, the so-called pizza model, did not survive into TEI P5, where all modules are considered equal even though there are some modules some of whose components are needed for almost any schema[iii]. A class is an abstract object to which elements point in order to express their semantic or structural status. In SGML-based versions of the Guidelines, classes were represented by parameter entities which could be modified at the document instance level, thus providing extension points for the encoding scheme. In RELAXNG, classes are represented as patterns, with similar capabilities.

At its simplest, a customization just specifies a number of modules. For almost every practical application of the TEI, this will over-generate, not only in the sense that the resulting schema will contain specifications for components that will never be used, but in that the TEI often provides multiple ways of encoding the same phenomenon, even in the same module. The TEI core module provides both `<bibl>` and `<biblStruct>` as ways of representing a bibliographic record; the same module provides a variety of elements designed to signal the function associated with visual distinctions such as italicisation or quote marks, while also providing a way of simply signalling the fact of visual salience or highlighting itself. With or without use of three quite different ways of representing the form that the visual salience actually takes in the source, provided by the attribute class att.global.rendition. Similarly, the TEI att.datable attribute class provides two distinct sets of attributes for normalising dates and times, one conforming to W3C, the other conforming to ISO. Plus, for good measure, a third sub-class called att.datable.custom which allows the user to specify their own conventions. The TEI is scrupulously agnostic even about how a TEI document itself is to be constructed: the classic TEI document comprises a TEI Header and a transcribed text; the transcribed text may however be combined with a set of digitized images, or replaced by one; it is also possible to replace (or complement) the traditional text transcription (which aims to capture the logical organization of the source document) with a ‘source-oriented’ transcription which captures just its physical organization eschewing other interpretive gestures. And there are plans to add a further text-level component to contain annotations made upon the text in a ‘standoff’ manner.

This multiplicity of choice can be bewildering and may seem absurd. Yet every element and attribute in the TEI Guidelines is there because some member of the scholarly community has plausibly argued that it is essential to their needs; where there is a choice, therefore, it is not because the TEI is indecisive, it is because all of the available options have been considered necessary by someone, even if no-one (except perhaps those blessed with the task of maintaining the TEI) ever considers all of them together.

A project wishing to use the TEI is therefore required, expected, advised to consider carefully how to use it. Simply selecting a few promising modules is not necessarily the best approach: you will also need to select from the components provided by those modules, since selecting everything available is a recipe for confusion. Those unwilling or inadequately resourced to make this effort can use one or other of the generic TEI customizations made available by the TEI itself (TEI Simple Print, for example), or by specific research communities (Epidoc is an excellent example). But it is my contention that adopting an off-the-peg encoding system is always going to be less satisfactory than customizing one that fits more precisely the actual needs of your project and the actual data you have modelled within it. (You did do a data analysis before you started, didn't you?).

And whether or not you did, it's painfully true that nothing in digital form is ever really finished. It's almost inevitable that as your project evolves, you will come across things you would do differently if you could start all over again. In the light of experience, you may well want to change the list of available elements to match more closely your actual encoding practices. Beginners often think that it's better to allow almost any kind of content in their schema: an extreme case of this misapprehension leads people to use TEI_all for everything. It may well be that your project started out a bit uncertain about the kind of data it would have to be able to handle. But as an encoding project matures, these uncertainties disappear and project-specific praxis becomes better understood. Surely it is better to have a schema that matches all and only the elements you now know you need, than one which allows anything? Every element you allow for in your schema is another element you need to explain to your encoders, document, find examples for, and check that it is being used consistently (if it is used at all). It's also another element that the poor over-worked software developer has to be prepared to handle. Reducing the number of elements permitted by your schema makes it easier for you to concentrate on the quality of your documentation, by introducing examples more appropriate to your project than those provided by vanilla TEI.

Similar considerations apply to attributes, and in particular to their range of values. At the outset you may not have been sure what values to permit for the *@foo* attribute on your `<bar>` elements, so you allowed anything. Now you have discovered that some of your encoders gave this attribute the value centre, others used center, and yet others used middle, all meaning (probably) the same thing. Now that you know which values you want, you will want to add a `<valList>` to your customization to enforce them, even if this entails some additional work cleaning up existing data.

Customization is thus first and foremost a matter of selection, or formally speaking a subsetting operation. However, as just indicated, a customization may also provide additional constraints for aspects of an encoding left underspecified by the TEI, which may not always amount to subsetting. For example, a customization which specifies that attribute values be taken from a closed list of possible values rather than being any token of the appropriate datatype is a subsetting operation: the set of documents considered valid by that customization is a pure subset of the set of documents considered valid by a schema lacking that particular customization. This is also true of a modification which changes the datatype of an attribute, for example from a string to an integer or a date, but not of the reverse modification, for example from date to string. A modification may also provide an alternative identifier for an element or an attribute, for example to translate its canonical English name into another language: this cannot be regarded as a subsetting operation without doing some violence to the meaning of the term. A modification may change the class memberships of an existing TEI element, so that it acquires attributes not previously available, or so that it may appear in contexts where it previously could not. A modification may change the content model of an element, so that it may contain different child elements, or so that its existing children may appear in a different order or with different cardinalities, and it may introduce additional constraints on the content of an element or the value of an attribute using rules not otherwise (or not readily) expressible in formal schema languages such as DTD or RELAXNG.

A user of the TEI is also at liberty to define entirely new elements, macros, classes or attributes, and to reference them from existing TEI components, within certain limits. New elements or attributes should be explicitly attached to a different, non TEI, namespace, and new elements should, as far as possible, be included in existing content models by making them members of existing TEI classes, rather than by explicitly modifying the content model of an existing element.

Despite this flexibility, there are still a few hard wired rules built into the TEI model, which the customizer ignores at their peril. For example, a TEI Header really *must* have a title statement (which must contain at least a title), a publication statement and a source description, even if the latter two have no significant content. A TEI `<text>` element really *must* contain a `<body>` element. TEI `<div>` elements really *must* nest correctly within one other. The structural classes in terms of which content models are defined really *must* be respected: hence one `<p>` cannot contain another, and a phrase level element such as `<hi>` cannot contain a block like element such as `<p>`.

Some of these rules are regularly debated on TEI forums, but for the most part they remain integral to the TEI model. It is a part of the definition of a TEI `<div>` that once you have encountered another nested `<div>` element within it, only div elements at the same hierarchic level are permitted until it finishes ; this is not to say that a non-tessellating division element might not be useful, but if one is defined it most be distinguished clearly from the existing TEI `<div>`, for example by placing it in a different namespace. Breaking these rules may have unexpected consequences. For example, a customization which removes the element `<title>` will result in a schema in which no TEI Header element can ever be considered valid, since the mandatory components of the TEI Header are an essential part of it; a TEI Header which lacks them is a different kind of object, and should not present itself as being something which it is not.

3. What is TEI conformance?

Umberto Eco remarks somewhere that a novel is a machine for generating interpretations. We might say that the TEI is a machine for generating schemas to formally represent such interpretations. However, just as not all interpretations of a novel have equivalent explanatory force, so the schemas generated by different TEI customizations may vary in their effectiveness or appropriateness. A schema represents a view of what it is meaningful to assert about a set of documents. As we have seen, a TEI schema does this with reference to the existing range of TEI specifications, selecting the particular distinctions it wishes to make, modifying some of them in ways which may or may not result in schema that is a subset of the full TEI, and possibly adding new categories of distinction entirely. It is reasonable to want to know how the original TEI schema has been modified, since this enables us to make comparisons amongst different customizations, to assess their relative closeness to the original Guidelines, and to determine what might be necessary to make documents using those different customizations interchangeable, if not interoperable. As Martin Holmes and others have pointed out, the pursuit of general unmediated interoperability amongst TEI documents is largely chimerical, whereas the information provided by a TEI customization will often be all that is needed to make them interchangeable.

The existing Chapter 23 of the TEI Guidelines introduces a notion of TEI conformance though it falls short of providing a full formal definition, either of what conformance means, or how it should be assessed. The following diagram is intended to demonstrate some of the notions discussed in that chapter, and elsewhere.

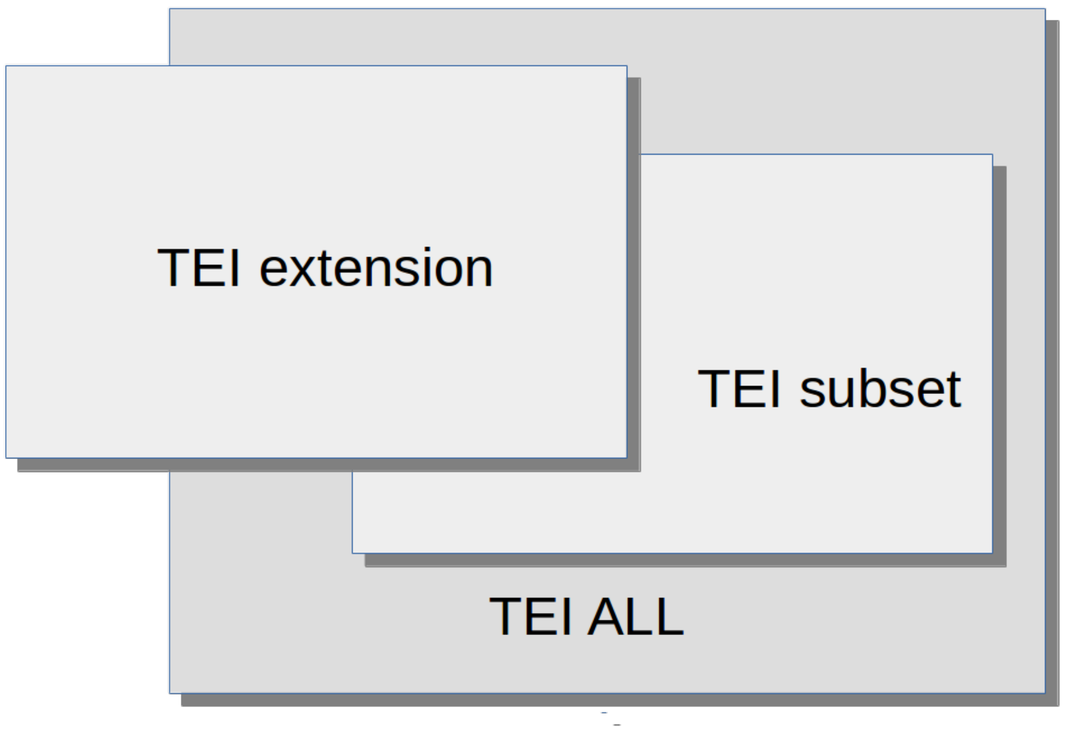


Figure 1.

Each of the shapes here may be understood to represent three different things:

- an ODD : that is, a collection of TEI specifications
- a formal schema generated from that ODD, and its natural language documentation
- the set of documents considered valid by that schema

The TEI provides a completely unmodified schema called tei_all : this contains all of the elements, classes, macros, etc. defined by the TEI. As noted above, for all practical purposes a user of the TEI must make a selection from this cornucopia, and I will call that selection a ‘TEI customization’. Of course there are many many possible TEI customizations, each involving different choices of elements or attributes or classes, but there are at least two different kinds of customization: a *TEI subset* and a *TEI extension*.

In some cases, the modifications result in a schema which regards as valid a subset of the documents considered valid by tei_all: I will call this a ‘TEI subset’. In others, this is not the case. Renaming operations, for example, cannot ever result in a TEI subset: a document in which the element names have all been changed to their French equivalents cannot be validated by an English language version of tei_all. Equally, a customization which adds new elements or attributes cannot ever result in a subset. A change to the content model or the class memberships of existing TEI elements may or may not result in a TEI subset. For example, if tei_all does not specify an order for the child elements of some content model, a modification which constrains that order will regard as valid a subset of the documents considered valid by tei_all. The reverse is not the case, however: if tei_all does specify an order, a modification which relaxes that constraint will result in a schema that considers valid some documents considered invalid by tei_all. I propose the term 'TEI extension' for any customization which results in a schema that considers as valid a set of documents which is not a proper subset of the documents regarded as valid by the tei_all schema.

It is natural that particular importance should be attached to validity against a schema, since this is something which can be automatically tested. But it would be a mistake to consider that validity against tei_all has any particular significance in assessing the status of a customization, other than to determine whether it is a TEI subset or a TEI extension. The TEI was designed to support either kind of customization, and either should be considered equally 'conformant', if that term is meant to imply something about coherence with the design goals or recommendations of the Initiative. Those recommendations do however include a variety of constraints about modifications, some of them well formulated and verifiable, others less so. For example, the TEI Guidelines require that a modification should use namespaces in a coherent manner: in particular, that elements not defined by the TEI, or TEI elements whose definition has been modified to such an extent that they are arguably no longer the same element should not be defined within the TEI namespace, but instead use a customization-specific namespace. Software such as onvdl can be used to check (within limits) the validity of a document using multiple namespaces[iv]. But the ability to extend the range of encodings supported by the TEI simply and straightforwardly remains a fundamental principle for a scheme which is intended to serve the needs of research. This principle has several important benefits:

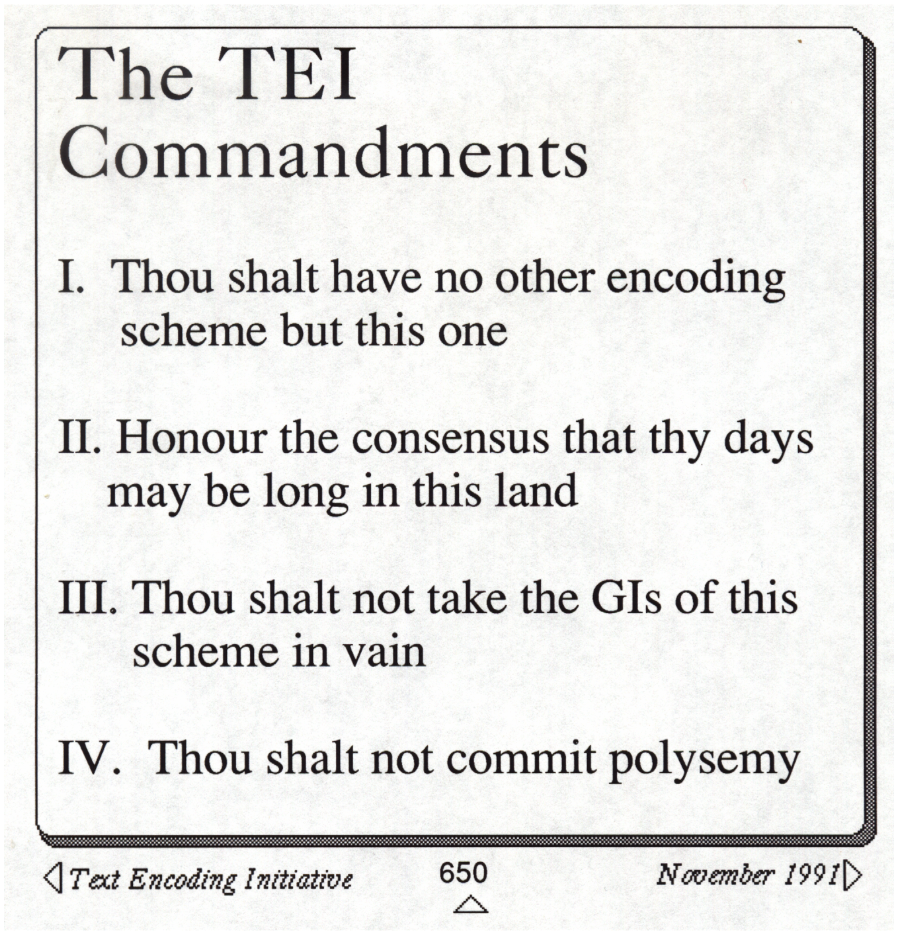
- it enables the TEI to leverage with comparative ease otherODD specialised XML vocabularies, such as MathML, SVG, or most recently MML;
- it facilitates and encourages the development of new TEI components by the broader community;
- it simplifies the task of interchange by reducing the possibility of ambiguous or incoherent encoding.

Even in the case of a customization which has eschewed extension and appears to be a straightforward TEI subset, an assessment of TEI conformance involves attention to some less easily verifiable constraints. In particular, I suggest, there are two important if largely unenforceable requirements of 'honesty' and 'explicitness'.

By 'honesty' I mean that elements in the TEI namespace must respect the semantics which the TEI Guidelines supply as a part of their definition. For example, the TEI defines an element `<l>` as containing 'a single, possibly incomplete, line of verse'. Assuming that your encoding makes a clear distinction between verse and prose, it would therefore be dishonest to use this element to mark line breaks in prose, since to do so would imply that the element contains verse rather than prose. Most TEI elements are provided in order to make an assertion about the semantics of a piece of text : that it contains a personal name rather than a place name, for example, or a date rather than a number. Misapplying such elements is clearly counter-productive. (Honestly made misreadings are of course entirely forgiveable: an encoding always asserts an interpretation, not the absolute truth of that interpretation)

By 'explicitness' I mean that all modifications should be properly documented, ideally by means of an ODD specifying exactly how the TEI declarations on which they are based have been derived. (An ODD need not of course be based on the TEI at all, but in that case the question of TEI conformance does not arise). The ODD language, as we have seen, is rich in documentary components, not all of which are automatically processable, if only because their processing is not fully specified (the `<equiv>` and `<altIdentifier>` elements for example). But it is usually much easier to determine how the markup of a set of documents should be interpreted or processed from an ODD than it is from the many pages of human-readable documentation which may be needed to explain everything about an idiosyncratic encoding scheme.

In 1991, I made an attempt to summarize the kind of conformance envisaged by what was then the newly published TEI P3. With the possible exception of the first, these four humorously presented 'TEI Commandments' have much in common with the more nuanced statement of what conformance should be that we might produce today.



The TEI Commandments

I. Thou shalt have no other encoding scheme but this one

II. Honour the consensus that thy days may be long in this land

III. Thou shalt not take the GIs of this scheme in vain

IV. Thou shalt not commit polysemy

Text Encoding Initiative 650 November 1991

Figure 2.

Today, I suggest that we should say of a document that it is 'TEI conformant' iff :

- it is a well formed XML document; and
- it is valid against one or more schemas, which may be either a TEI subset or a TEI extension; and
- its usage of elements in the TEI namespace is compatible with the intended function of those elements as defined by the TEI Guidelines; and
- its usage of the TEI markup scheme is fully described by a TEI-conformant ODD or analogous documentation.

The purpose of these rules is to make interchange of documents easier. They do not guarantee it, and they certainly do not provide any guarantee of interoperability. But they make much simpler for example the kind of scenario envisaged by Holmes 2016 in which a richly encoded highly personalised TEI encoding can be simply down-translated to other, possibly less expressive, semi-standardized encodings for purposes of interchange. As more and more independent agencies undertake mass digitization and encoding projects, the risk of a new confusion of tongues -- the threatened Tower of Babel which the TEI was specifically created to resist -- has not retreated. A definition of conformance which relies on an enforced lowest common denominator standard (Dublin Core springs to mind) makes it hard to benefit from truly sophisticated and scholarly standards. One which promotes permissiveness and extensibility, as the TEI does, has to balance the sophistication of what it makes feasible with a clear and accessible definition of its markup. Unlike many other standards, the goal of the TEI 'standard' is not to enforce consistency of encoding, but to provide a means by which encoding choices and policies may be more readily understood, and hence more easily made algorithmically comparable.

Notes

i. Earlier versions of the TEI did not attempt to duplicate the features of existing schema languages, but simply embedded expressions in the DTD language (in versions prior to P4), or in RELAXNG schema language (in TEI P4

and early version of P5). From TEI P5 version 3.0 however, the XML vocabulary in which the TEI is written was expanded to include facilities for expressing content models directly in TEI. (see Burnard 2013). One motivation for this was to reduce the dependence of the TEI documentation scheme on other modelling langages, in line with the priorities expressed in the TEI's foundational design goals (see TEI 1988), according to which ‘The Text Encoding Initiative will develop a conforming SGML application, if it can meet the needs of researchers by doing so. Where research needs require constructs unavailable with SGML, however, research must take precedence over the standard’.
ii. Rahtz & Burnard 2004 introduces this ‘One Document Does it all’ system as currently instantiated.
iii. The tei module supplies declarations for generally useful macros, datatypes, model and attribute classes; the core module which supplies declarations for elements needed in ‘all kinds of document’; the header module which supplies metadata elements; and the textstructure module which supports the basic organization of book like objects, for example.
iv. onvdl routes different parts of an XML document for validation against possibly many different schemas, using the Namespace-based Validsation Despatching Language, defined as part 4 of ISO 19757

[1]. 2004 Burnard, Lou and Sebastian Rahtz: *RelaxNG with Son of ODD*. Extreme Markup Languages.

[2]. *Resolving the Durand Conundrum*, Journal of the Text Encoding Initiative[Online], Issue 6 | December 2013, URL : http://jtei.revues.org/842 ; DOI : 10.4000/jtei.842

[3]. *Design principles for text encoding guidelines* (TEI ED P1, 1988, revised 1990)

[4]. Holmes, Martin *Whatever happened to interchange?* Digital Scholarship in the Humanities, 2016.